ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ
Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп'ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
_____ Олена ОЛЬХОВСЬКА
(підпис)

«___» _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему
«Програмна реалізація системи контролю знань за модулем "Моделювання об’єктів" з дисципліни «Моделювання об’єктів та процесів»»

зі спеціальності 122 «Комп’ютерні науки»
освітня програма «Комп’ютерні науки»
ступеня бакалавра

Виконавець роботи Почерняєв Дмитро Андрійович
_____ «___» _____ 2025 р.
(підпис)

Науковий керівник к.ф.-м.н., доц., Чілікіна Тетяна Василівна
_____ «___» _____ 202_ р.
(підпис)

Рецензент _____

ПОЛТАВА 2025

Вступ

В епоху стрімкого розвитку інформаційних технологій, ефективне навчання та оцінювання знань набувають особливого значення. Одним із ключових аспектів сучасної освіти є моделювання об'єктів та процесів, що дозволяє студентам глибше зрозуміти складні системи та явища. У зв'язку з цим, розробка ефективних систем тестування знань у цій галузі стає нагальною потребою.

Ця дипломна робота присвячена проектуванню елементів системи тестування знань за модулем "Моделювання об'єктів" з дисципліни "Моделювання об'єктів та процесів". Метою роботи є створення інструменту, який дозволить об'єктивно оцінити рівень засвоєння студентами теоретичних знань та практичних навичок у цій галузі.

Актуальність теми зумовлена необхідністю підвищення якості освіти у сфері інформаційних технологій. Ефективна система тестування знань дозволить не лише оцінити рівень засвоєння матеріалу, але й виявити прогалини у знаннях студентів, що сприятиме їхньому подальшому навчанню.

Мета проєкту – Алгоритмізація та програмування тренажера з теми «Числові послідовності" дистанційного навчального курсу «Математичний аналіз».

Об’єктом розробки є програмна реалізація навчального тренажеру з теми «Числові послідовності» дистанційного навчального курсу «Математичний аналіз», блок-схема, алгоритм.

Предмет розробки — навчальний тренажер з теми «Числові послідовності».

Перелік використаних методів: при створенні програми-тренажера було використано мову програмування C++;

Пояснювальна записка складається з чотирьох розділів. У першому розділі розглянуто постановку задачі. У другому розділі було оглянуто подібні програмні продукти та виділено позитивні та негативні аспекти. У третьому

розділі представлено огляд матеріалу з теми, алгоритм роботи тренажера та блок-схеми роботи тренажера. У четвертому розділі розглянуто якими методами створювався тренажер та яке програмне середовище було обрано для розробки.

.

Обсяг пояснювальної записки: 37 стор., в т.ч. основна частина - 25 стор., додатки - 1 стор., рис. 11, джерела - 8 назв.

Урахування сучасних тенденцій у навчанні та використання технологій дозволяють ефективно поєднувати традиційне викладання з дистанційними форматами. Створений тренажер дозволяє оптимізувати навчальний процес, надаючи можливість студентам отримати доступ до ресурсів для вивчення числових послідовностей у будь-який зручний для них час та місце.

Враховуючи сучасні вимоги до методів навчання, створений програмний продукт інтегрується в освітній процес, сприяючи підвищенню якості викладання математики.

Активне використання тренажера у навчальних закладах підсилює актуальність його внеску у сучасну освітню практику. Навички математичного аналізу та робота з числовими послідовностями є критичними для студентів у вивченні математики.

У рамках цієї роботи буде розглянуто основні принципи моделювання об'єктів, а також методи та алгоритми, що використовуються для створення моделей. Особливу увагу буде приділено проектуванню структури системи тестування, розробці тестових завдань та алгоритмів оцінюання результатів.

Для реалізації проекту буде використано мови програмування що дозволить створити ефективну та надійну систему тестування. Результати роботи можуть бути використані для організації навчального процесу у вищих навчальних закладах, а також для самостійної підготовки студентів.

РОЗДІЛ 1.ПОСТАНОВКА ЗАДАЧІ

В рамках кваліфікаційної роботи потрібно розробити програмне забезпечення для контролю знань з простим та зручним дизайном, а також зрозумілим функціоналом.

Розв’язання задач із використанням комп’ютера характеризується декількома етапами, частина з яких виконуються безпосередньо людиною, а решта — людиною і машиною:

- Постановка задачі;
- Опис початкових даних, формулювання мети задачі;
- Побудова інформаційної моделі;
- Опис реального об’єкта дослідження в припустимих для реалізації задачі термінах, щоб звести дослідження реального об’єкта до розв’язання задачі на моделі;
- Вибір програмного забезпечення;
- Визначення необхідного прикладного програмного забезпечення;
- Аналіз створеної програми та її тестування і.

Питання повинні бути різноманітними та охоплювати всі теми.

Розділ 1. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Основні поняття та принципи моделювання об'єктів та процесів.

Моделювання об'єктів та процесів є фундаментальним інструментом для дослідження та розуміння складних систем. В основі моделювання лежить принцип заміни реального об'єкта або процесу його спрощеним, але інформативним представленням – моделлю. Ця модель дозволяє аналізувати поведінку системи, прогнозувати її реакцію на різні впливи та оптимізувати її характеристики.

Моделювання є ключовим інструментом для розуміння та аналізу складних систем. Існує кілька типів моделей, кожен з яких має свої особливості та застосування.

Математичні моделі використовують мову математики для опису поведінки системи. Вони складаються з рівнянь, формул та інших математичних виразів, які відображають зв'язки між різними параметрами системи. Переваги: математичні моделі дозволяють точно аналізувати кількісні характеристики системи; вони можуть використовуватися для прогнозування поведінки системи в різних умовах; математичні моделі забезпечують високий рівень абстракції, що дозволяє зосередитися на ключових аспектах системи. Обмеження: розробка математичних моделей може бути складною, особливо для складних систем; інтерпретація результатів математичного моделювання може вимагати спеціальних знань; математичні моделі можуть не враховувати всі фактори, що впливають на поведінку системи.

Фізичні моделі відтворюють фізичні властивості реального об'єкта або процесу. Вони можуть бути виконані з різних матеріалів та використовуватися для експериментального дослідження поведінки системи. Переваги: фізичні моделі дозволяють наочно продемонструвати поведінку системи; вони можуть використовуватися для експериментального дослідження впливу різних факторів на систему; фізичні моделі можуть бути корисними для навчання та

демонстрації. Обмеження: виготовлення фізичних моделей може бути дорогим та складним; фізичні моделі можуть неточно відтворювати поведінку реальної системи; експерименти з фізичними моделями можуть бути обмежені в часі та ресурсах.

Комп'ютерні моделі використовують комп'ютерні програми для імітації поведінки системи. Вони дозволяють швидко та ефективно аналізувати складні системи, використовуючи різні алгоритми та методи. Переваги: комп'ютерні моделі дозволяють швидко та ефективно аналізувати складні системи; вони можуть використовуватися для моделювання поведінки системи в різних умовах; комп'ютерні моделі дозволяють візуалізувати результати моделювання. Обмеження: розробка комп'ютерних моделей вимагає навичок програмування; точність комп'ютерних моделей залежить від точності алгоритмів та даних; комп'ютерні моделі можуть бути складними для верифікації та валідації.

У контексті вашої курсової роботи, присвяченої проектуванню системи тестування знань з моделювання об'єктів та процесів, комп'ютерні моделі мають особливе значення. Використання мов програмування C/C++ дозволить вам створити ефективну та надійну систему, яка зможе оцінити знання студентів у цій галузі..

Процес моделювання складається з кількох етапів. На першому етапі визначається мета моделювання та вибирається тип моделі. На другому етапі розробляється модель, що включає опис структури системи, її параметрів та зв'язків між ними. На третьому етапі модель перевіряється на адекватність, тобто на відповідність реальному об'єкту або процесу. На четвертому етапі модель використовується для аналізу поведінки системи та прийняття рішень.

Моделювання широко застосовується в різних галузях, включаючи інженерію, фізику, хімію, біологію, економіку, соціальні науки та інформаційні технології. В інженерії моделювання використовується для проектування та аналізу складних технічних систем, таких як літаки, автомобілі та мости. У фізиці моделювання використовується для дослідження фундаментальних законів природи, таких як закони руху та електромагнетизму. В економіці

моделювання використовується для прогнозування економічних показників та аналізу впливу різних факторів на економіку. В інформаційних технологіях моделювання використовується для розробки та аналізу комп'ютерних систем, таких як бази даних та мережі.

2.2. Методи та технології тестування знань.

Тестування знань є невід'ємною частиною оцінювання навчальних досягнень студентів. Воно дозволяє об'єктивно оцінити рівень засвоєння матеріалу, виявити прогалини у знаннях та надати зворотний зв'язок для подальшого навчання. Існує безліч методів та технологій тестування, кожен з яких має свої переваги та недоліки.

Тестування знань є важливим інструментом для оцінювання рівня засвоєння матеріалу студентами. Залежно від мети та змісту, тести можна класифікувати на кілька видів.

Теоретичні тести призначені для перевірки знань теоретичних концепцій, принципів та визначень. Вони оцінюють розуміння студентами основних положень дисципліни та їх здатність до аналізу та синтезу теоретичного матеріалу. У контексті курсової роботи, теоретичні тести можуть включати питання, що перевіряють знання основних принципів моделювання об'єктів та процесів, видів моделей, етапів моделювання, а також математичних методів, що використовуються в моделюванні. Наприклад, студентам можуть бути запропоновані питання на визначення основних понять моделювання, опис переваг та недоліків різних видів моделей, або пояснення етапів побудови моделі.

Практичні тести призначені для перевірки вміння студентів застосовувати теоретичні знання на практиці. Вони оцінюють здатність студентів вирішувати практичні завдання, використовувати інструменти та методи моделювання, а також інтерпретувати результати моделювання. У контексті роботи, практичні тести можуть включати завдання на побудову математичних моделей

конкретних об'єктів або процесів, розробку комп'ютерних моделей з або аналіз результатів моделювання. Наприклад, студентам можуть бути запропоновані завдання на створення математичної моделі руху автомобіля, розробку програми для імітації роботи системи обслуговування, або аналіз результатів моделювання поширення тепла в твердому тілі.

Комбіновані тести поєднують теоретичні та практичні завдання. Вони оцінюють як знання теоретичних концепцій, так і вміння застосовувати їх на практиці. У контексті курсової роботи, комбіновані тести можуть включати питання на пояснення теоретичних принципів моделювання та практичні завдання на побудову та аналіз моделей. Наприклад, студентам можуть бути запропоновані питання на пояснення принципів побудови математичних моделей та завдання на розробку математичної моделі конкретного об'єкта.

Вибір виду тесту залежить від мети тестування та специфіки навчального матеріалу. Для забезпечення всебічної та об'єктивної оцінки знань студентів доцільно використовувати комбінований підхід, що поєднує різні види тестів..

Формати тестових завдань також різноманітні. Завдання з множинним вибором передбачають вибір одного або кількох правильних варіантів відповідей з кількох запропонованих. Вони легко перевіряються та дозволяють охопити широкий спектр тем. Відкриті питання вимагають від студентів самостійно формулювати відповіді. Вони дозволяють оцінити глибину розуміння матеріалу та вміння логічно мислити. Завдання на відповідність передбачають встановлення відповідності між елементами двох множин. Вони часто використовуються для перевірки знання термінології та класифікацій.

Критерії оцінювання результатів тестування повинні бути чіткими та об'єктивними. Вони можуть базуватися на кількості правильних відповідей, якості відповідей на відкриті питання, повноті та точності встановлення відповідностей. Важливо також враховувати складність завдань та час, відведений на їх виконання. Для забезпечення об'єктивності оцінювання можна використовувати статистичні методи, такі як розрахунок середнього балу, стандартного відхилення та коефіцієнта кореляції.

Вибір методів та технологій тестування, форматів тестових завдань та критеріїв оцінювання залежить від мети тестування, специфіки навчального матеріалу та доступних ресурсів. Важливо використовувати комплексний підхід, що поєднує різні методи та формати, для забезпечення всебічної та об'єктивної оцінки знань студентів.

1.3. Огляд існуючих систем тестування знань.

Сучасний ринок пропонує широкий спектр систем тестування знань, кожна з яких має свої особливості та призначення. Серед найбільш поширених можна виділити системи дистанційного навчання (LMS), онлайн-платформи для тестування, а також спеціалізовані програми для створення та проведення тестів.

Системи дистанційного навчання, такі як Moodle, Canvas та Blackboard, надають комплексні рішення для організації навчального процесу, включаючи інструменти для створення та проведення тестів. Їх перевагами є широка функціональність, можливість інтеграції з іншими навчальними ресурсами та зручний інтерфейс для користувачів. Однак, вони можуть бути складними в налаштуванні та потребувати значних ресурсів для підтримки.

Онлайн-платформи для тестування, такі як Google Forms, Quizizz та Kahoot!, пропонують прості та зручні інструменти для створення та проведення тестів. Їх перевагами є доступність, швидкість створення тестів та можливість отримання миттєвих результатів. Однак, вони можуть мати обмежену функціональність та не підходити для складних тестів з різними типами завдань.

Спеціалізовані програми для створення та проведення тестів, такі як TestGen та Respondus, надають розширені можливості для створення та налаштування тестів. Їх перевагами є гнучкість, можливість створення складних тестів з різними типами завдань та можливість інтеграції з іншими

навчальними системами. Однак, вони можуть бути платними та потребувати спеціальних навичок для використання.

При виборі технологій для реалізації власної системи тестування необхідно враховувати мету тестування, специфіку навчального матеріалу, доступні ресурси та навички програмування. Для реалізації системи тестування знань за модулем "Моделювання об'єктів" з дисципліни "Моделювання об'єктів та процесів" доцільно використовувати мови оскільки вони дозволяють створити ефективну та надійну систему з широкими можливостями для налаштування та інтеграції з іншими системами.

Для зберігання даних можна використовувати бази даних, такі як SQLite або MySQL. Для створення інтерфейсу користувача можна використовувати бібліотеки, такі як Qt або wxWidgets. Для створення веб-інтерфейсу можна використовувати фреймворки, такі як Django або Flask.

Вибір конкретних технологій залежить від вимог до системи та навичок програміста. Важливо використовувати комплексний підхід, що поєднує різні технології, для забезпечення ефективної та надійної роботи системи тестування.

РОЗДІЛ 3 ТЕОРЕТИЧНА ЧАСТИНА

Проектування елементів системи тестування знань

2.1. Визначення вимог до системи.

У рамках проектування системи тестування знань, я приділив особливу увагу функціональності, пов'язаній з проходженням тестів користувачами та автоматичним оцінюванням результатів. Ця функціональність є ключовою, оскільки вона забезпечує основну мету системи – оцінювання рівня знань користувачів.

Процес проходження тесту повинен бути максимально зручним та інтуїтивно зрозумілим для користувача. Після запуску тесту, користувачеві послідовно відображаються питання, кожне з яких містить чотири варіанти відповіді. Користувач має можливість вибрати один з варіантів, використовуючи клавіатуру або мишу. Система повинна забезпечувати можливість навігації між питаннями, щоб користувач міг повернутися до попередніх питань та змінити свої відповіді, якщо це необхідно.

Після того, як користувач відповів на всі питання, система автоматично обробляє результати. Для кожного питання система порівнює відповідь користувача з правильним варіантом, який зберігається в базі даних. На основі цього порівняння система визначає, чи була відповідь правильною.

Після обробки відповідей на всі питання, система обчислює оцінку користувача. Оцінка може бути виражена у вигляді відсотка правильних відповідей або у вигляді бальної оцінки. Система повинна забезпечувати можливість налаштування критеріїв оцінювання, щоб адміністратор міг визначити, як оцінювати результати тестів.

Після обчислення оцінки, система надає користувачеві зворотний зв'язок. Зворотний зв'язок може включати інформацію про правильні та неправильні відповіді, а також детальну інформацію про кожне питання та правильну

відповідь. Це дозволяє користувачеві зрозуміти, де він допустив помилки, та покращити свої знання.

Окрім цього, система повинна забезпечувати можливість збереження результатів тестів у базі даних. Це дозволяє адміністратору аналізувати результати тестів, виявляти слабкі місця у знаннях користувачів та приймати рішення щодо покращення навчального процесу.

Окрім функціональних вимог, я також врахував нефункціональні вимоги, які є не менш важливими для успішної роботи системи. Серед них – зручність використання, надійність та безпека.

Зручність використання є ключовим фактором, що визначає, наскільки легко та ефективно користувачі зможуть взаємодіяти з системою. Інтуїтивно зрозумілий інтерфейс, легкість навігації та швидкість виконання операцій є критично важливими для забезпечення комфортного досвіду користувача. Інтерфейс повинен бути розроблений таким чином, щоб користувачі могли легко знаходити необхідні функції та виконувати завдання без зайвих зусиль. Навігація між сторінками та розділами системи повинна бути логічною та простою. Швидкість виконання операцій повинна бути достатньою для того, щоб користувачі не відчували затримок та очікувань.

2.2. Розробка структури бази даних для зберігання тестових завдань та результатів тестування.

Для ефективного зберігання та обробки даних, необхідних для роботи системи тестування знань, я розробив реляційну базу даних. Ця структура є фундаментом для надійного та швидкого доступу до інформації, необхідної для функціонування системи.

Основною таблицею є "Тести", яка містить загальну інформацію про кожен тест. Вона включає такі поля, як "ідентифікатор тесту", "назва тесту", "опис тесту", "тривалість тесту" та "критерії оцінювання". "Ідентифікатор тесту" є унікальним ключем, який дозволяє однозначно ідентифікувати кожен

тест. "Назва тесту" та "опис тесту" надають інформацію про зміст та призначення тесту. "Тривалість тесту" визначає час, відведений на проходження тесту. "Критерії оцінювання" описують, як будуть оцінюватися результати тестування.

Таблиця "Питання" містить інформацію про кожне питання тесту. Вона включає такі поля, як "ідентифікатор питання", "текст питання", "тип питання" та "складність питання". "Ідентифікатор питання" є унікальним ключем, який дозволяє однозначно ідентифікувати кожне питання. "Текст питання" містить формулювання питання. "Тип питання" визначає тип питання, наприклад, "з вибором одного варіанту" або "з вибором кількох варіантів". "Складність питання" визначає рівень складності питання.

Таблиця "Варіанти відповідей" містить інформацію про варіанти відповідей на кожне питання. Вона включає такі поля, як "ідентифікатор варіанту відповіді", "текст варіанту відповіді" та "ідентифікатор питання". "Ідентифікатор варіанту відповіді" є унікальним ключем, який дозволяє однозначно ідентифікувати кожен варіант відповіді. "Текст варіанту відповіді" містить формулювання варіанту відповіді. "Ідентифікатор питання" зв'язує варіант відповіді з відповідним питанням.

Таблиця "Результати" містить інформацію про результати проходження тестів користувачами. Вона включає такі поля, як "ідентифікатор результату", "ім'я користувача", "ідентифікатор тесту", "дата та час проходження", "відповіді на питання" та "отримана оцінка". "Ідентифікатор результату" є унікальним ключем, який дозволяє однозначно ідентифікувати кожен результат. "Ім'я користувача" визначає користувача, який проходив тест. "Ідентифікатор тесту" зв'язує результат з відповідним тестом. "Дата та час проходження" визначають час, коли користувач проходив тест. "Відповіді на питання" містять інформацію про відповіді користувача на кожне питання. "Отримана оцінка" визначає оцінку, отриману користувачем за проходження тесту.

Ця структура бази даних дозволяє ефективно зберігати та отримувати дані, необхідні для роботи системи тестування знань. Вона забезпечує надійне

зберігання інформації про тести, питання, варіанти відповідей та результати тестування.

2.3. Проектування інтерфейсу користувача.

При проектуванні інтерфейсу користувача я керувався принципами зручності та інтуїтивності, щоб забезпечити максимально комфортний досвід для користувачів. Інтерфейс складається з двох основних частин: інтерфейсу для проходження тестів та інтерфейсу для перегляду результатів.

Інтерфейс для проходження тестів розроблений таким чином, щоб користувач міг легко зосередитися на питаннях та варіантах відповідей. На екрані відображається текст питання, а під ним – чотири варіанти відповіді. Користувач може вибрати один з варіантів, натиснувши на відповідну кнопку або використовуючи клавіатуру. Для зручності навігації між питаннями передбачені кнопки "Попереднє питання" та "Наступне питання". У верхній частині екрана відображається номер поточного питання та загальна кількість питань у тесті. Також відображається таймер, який показує час, що залишився до завершення тесту. Після відповіді на останнє питання користувач може натиснути кнопку "Завершити тест", щоб завершити проходження тесту.

Інтерфейс для перегляду результатів надає користувачеві можливість детально ознайомитися з результатами проходження тесту. На екрані відображається список питань, на які користувач відповів. Для кожного питання відображається текст питання, вибраний користувачем варіант відповіді, правильний варіант відповіді та статус відповіді (правильно/неправильно). У верхній частині екрана відображається загальна оцінка, отримана користувачем за проходження тесту. Також відображається час, витрачений на проходження тесту.

Інтерфейс розроблений з використанням сучасних принципів дизайну, що забезпечує його зручність та ефективність. Використані кольори та шрифти

підібрані таким чином, щоб забезпечити комфортне сприйняття інформації. Розташування елементів інтерфейсу є логічним та інтуїтивно зрозумілим.

Для реалізації інтерфейсу користувача я планую використати бібліотеку Qt, яка надає широкі можливості для створення графічних інтерфейсів.

2.4. Вибір мови програмування та інструментів розробки.

При виборі мови програмування та інструментів розробки я керувався принципами ефективності, надійності та зручності. Для реалізації системи тестування знань я обрав мову програмування

Використання цих інструментів дозволить мені створити ефективну та надійну систему тестування знань. C/C++ забезпечить швидкість та надійність роботи системи, Qt забезпечить зручний та сучасний інтерфейс користувача, а SQLite забезпечить надійне зберігання даних.

РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА

4.1. Розробка загальної структури системи

Першим етапом реалізації освітнього веб-додатку стала побудова його загальної структури. Вона охоплює **логічну** та **фізичну архітектуру**, що визначають принципи організації елементів, їхню взаємодію, а також фізичне розміщення файлів і ресурсів.

Логічна структура описує, які функціональні модулі містить система, як вони між собою пов’язані та яку роль виконують у загальному процесі навчання.

Система складається з таких основних логічних блоків:

- Інтерфейс користувача (UI) — забезпечує навігацію, виведення теоретичного матеріалу, запуск тестів, роботу з довідником.
- Навчальний модуль — реалізує подання теоретичного контенту у вигляді розділів, карток, таймлайнів та схем.
- Тестовий модуль — відповідає за формування запитань, перевірку відповідей, обчислення балів, відображення статистики.
- Модуль збереження даних — локально зберігає прогрес користувача, результати тестування, закладки та нотатки за допомогою технології LocalStorage.
- Глосарій — реалізує швидкий доступ до термінології, пошук, фільтрування та перегляд прикладів.

Взаємозв'язки між модулями реалізовані через JavaScript-функції, що реагують на дії користувача (натискання кнопок, навігація, пошук тощо) і оновлюють відповідні частини інтерфейсу без перезавантаження сторінки.

Схема логічної структури (див. рис 4.1):

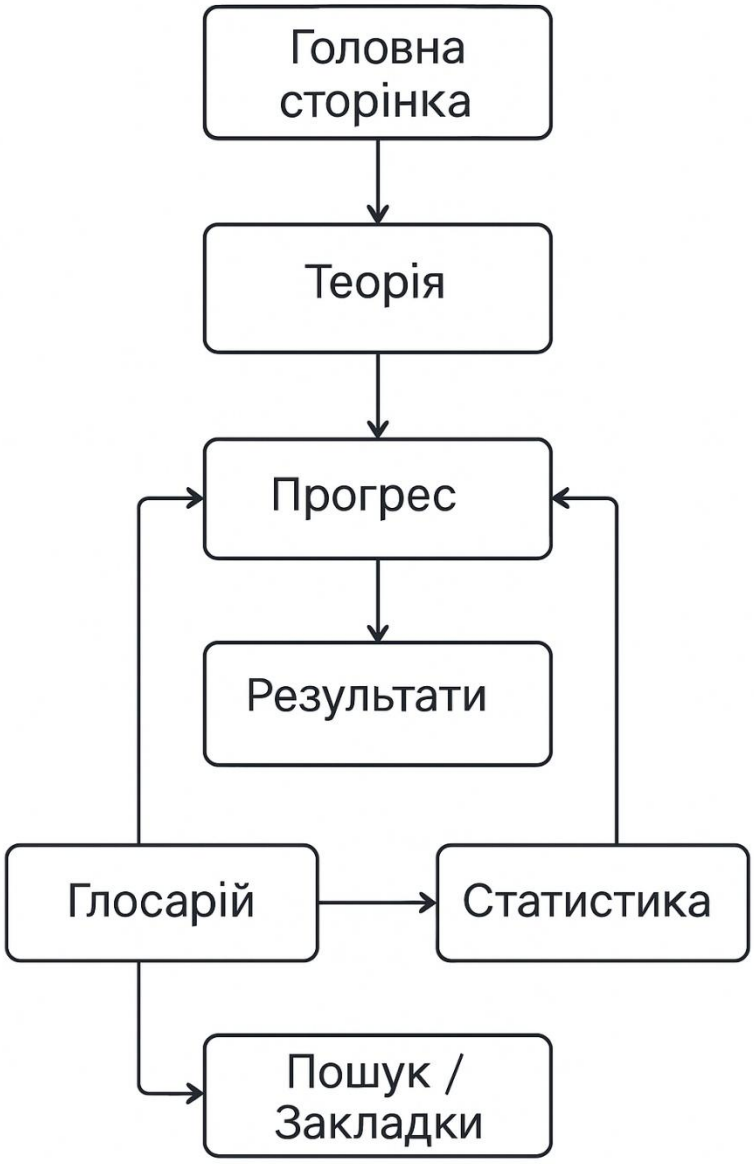

Рисунок 4.1 – Діаграма логічної структури сайту

Фізична структура відображає спосіб організації проєкту на рівні файлів і каталогів. Додаток реалізовано як статичний SPA-застосунок, що повністю працює у браузері користувача без потреби в серверній обробці.

Структура проєкту:

```
│
├── index.html        # розмітка всіх модулів
├── styles.css        # Всі стилі для інтерфейсу, адаптивності, анімацій
├── script.js         # навігація, тести, глосарій, збереження
```

Кожен файл відповідає за свою область:

- index.html — містить усі логічні секції, які відображаються або ховаються через JavaScript.
- styles.css — відповідає за візуальне оформлення: сітки, кольори, шрифти, ефекти.
- script.js — реалізує повну бізнес-логіку: обробку тестів, навігацію, роботу з LocalStorage, пошук у глосарії тощо.

Такий підхід дозволив реалізувати просту, ефективну й автономну систему, яка не потребує підключення до сервера. Користувач має можливість зберігати весь свій прогрес локально, що робить додаток швидким і придатним навіть для офлайн-роботи. Крім того, структура є масштабованою — до неї легко можна додати нові розділи, тести або функції без повної переробки проєкту.

4.2. Реалізація інтерфейсу користувача

Інтерфейс користувача є критично важливою складовою освітньої системи, адже саме через нього користувач взаємодіє з усіма модулями платформи: вивчає теоретичний матеріал, проходить тести, переглядає результати, працює з глосарієм. Під час реалізації було дотримано принципів зручності, доступності та адаптивності.

Основна навігація реалізована як фіксоване верхнє меню, що дозволяє миттєво переходити між основними розділами сайту: "Головна", "Теорія", "Тестування", "Довідник". При скролі сторінки меню залишається доступним для користувача.

Всі елементи інтерфейсу організовано у вигляді адаптивної сітки, що забезпечує коректне відображення на різних пристроях: десктопах, планшетах і

мобільних телефонах. Широко використовується сучасна візуальна мова: великі відступи, плавні анімації, яскраві кнопки, іконки Font Awesome.

Особливості реалізації:

- 3D-анімований куб на головній сторінці привертає увагу та підкреслює технологічність системи.
- Карткові блоки для тем теорії та тестів спрощують сприйняття інформації.
- Модальні вікна використовуються для виводу результатів та пояснень.
- Прогрес-бари відображають поточний стан навчання.
- Підказки та індикатори полегшують навігацію для нових користувачів.

Колірна палітра підібрана так, щоб поєднувати сучасність із академічністю: відтінки синього та бірюзового створюють відчуття довіри, спокою та фокусованості.

Шрифт Inter (від Google Fonts) з різними вагами забезпечує читабельність та ієрархію тексту. Для кнопок, заголовків та списків обрано великі розміри та чіткий контраст, що особливо важливо при використанні з мобільних пристроїв.

Інтерфейс також враховує основи доступності (a11y):

- Клавіатурна навігація
- Контрастні кольори
- Семантична HTML-структура
- Адаптація під screen-reader-и

Графічна реалізація

На рисунку нижче представлено макет користувацького інтерфейсу (див. рис. 4.2):

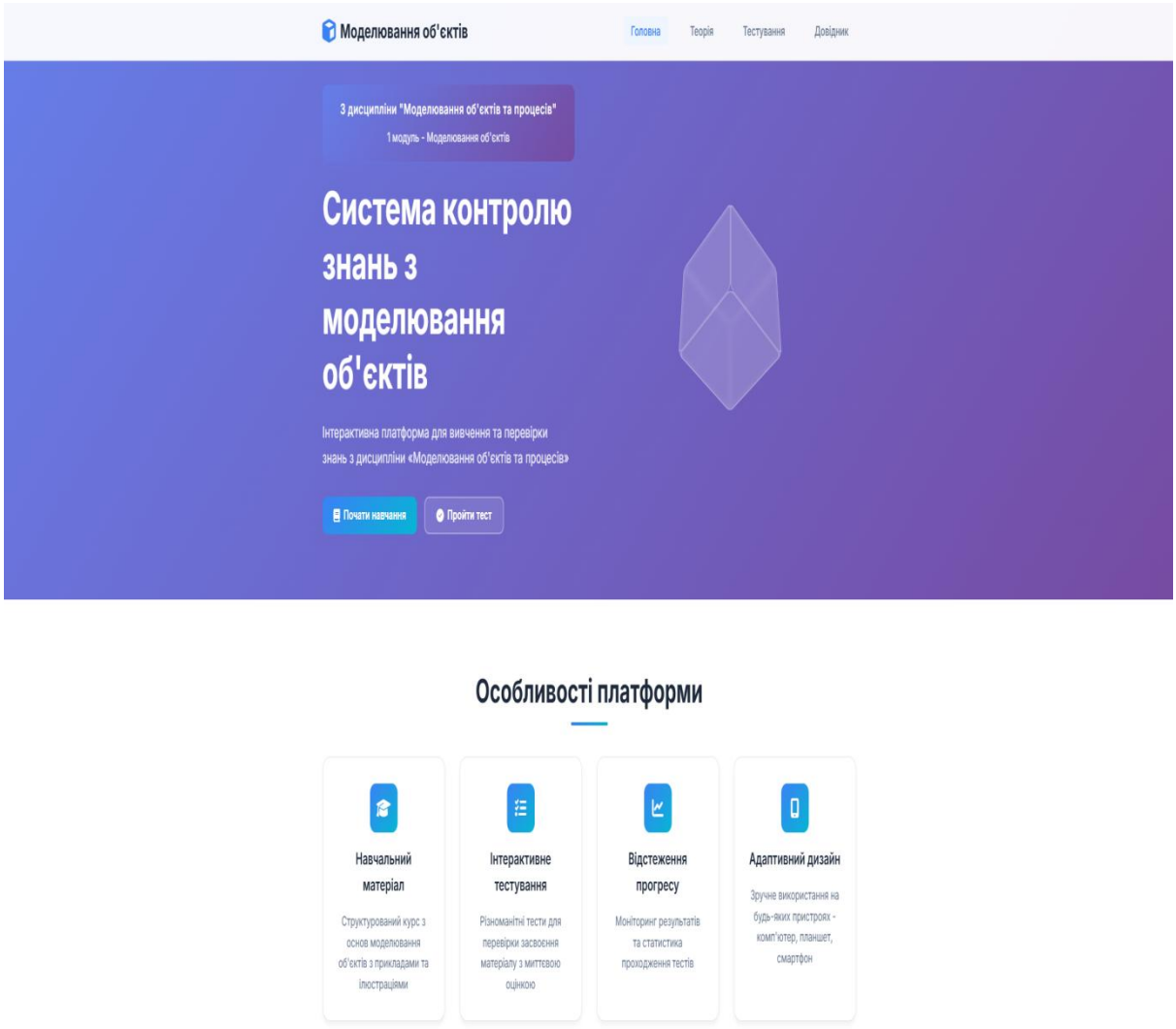

Рисунок 4.2 – Макет головної сторінки

4.3. Розробка модуля теоретичних матеріалів

Модуль теоретичних матеріалів є основою навчальної частини системи контролю знань. Його метою є подання ключових понять, моделей, підходів та прикладів, що формують базу знань студента з дисципліни «Моделювання об’єктів та процесів». У межах цього модуля реалізовано інтерактивну подачу інформації, що поєднує візуальні, текстові й логічно структуровані блоки (див. рис. 4.3).

Рисунок 4.3 – Схема теоретичної частини

Уся теорія поділена на дев’ять розділів, які охоплюють основні теми модуля "Моделювання об’єктів". Кожен розділ представлено у вигляді контент-картки, яку можна розгорнути для детального ознайомлення. Структура подачі дозволяє швидко орієнтуватися в матеріалі, вибираючи розділи відповідно до навчального плану.

Основні принципи розробки модуля:

- Структурованість — кожна тема розбита на підтеми, які містять визначення, приклади, порівняння.
- Інтерактивність — передбачено елементи розгортання, навігації, підсвічування термінів.

- Візуалізація — використано анімовані таймлайни, таблиці, діаграми (наприклад, UML).
- Навігаційна зручність — автоматична підсвітка активного розділу та збереження прогресу.

Кожен теоретичний розділ реалізований як окрема JS-функція, яка завантажується в центральну частину інтерфейсу при виборі відповідної теми. Для кращої організації та орієнтування, передбачено зміст розділів та поступове розкриття матеріалу (progressive disclosure), що дозволяє не перевантажувати користувача інформацією.

Теоретичні теми охоплюють як базові концепції, так і сучасні інструменти:

- Основи моделювання та класифікація моделей;
- Об’єктно-орієнтоване мислення та принципи ООП;
- Патерни проєктування та приклади їх використання;
- Життєвий цикл моделі — від аналізу до впровадження;
- UML-нотація та діаграми (класи, активності, стани тощо).

Для покращення запам’ятовування додано практичні приклади з реальних галузей, наприклад: моделювання виробничого процесу, проєктування комп’ютерної системи або створення логістичної моделі.

Збереження прогресу

Модуль інтегровано з системою локального збереження, що фіксує, які розділи були переглянуті, і дозволяє користувачу повернутись до незавершених тем.

Завдяки адаптивній верстці, теоретичні матеріали коректно відображаються як на великих екранах, так і на мобільних пристроях, що дозволяє використовувати систему на ходу.

4.4. Розробка системи тестування

Система тестування у веб-додатку реалізована як окремий інтерактивний модуль, призначений для перевірки рівня знань користувача з тем модуля «Моделювання об’єктів». Її структура і логіка побудовані таким чином, щоб забезпечити зручний, інтуїтивний та ефективний процес проходження тестів, а також зберігати результати користувача для подальшого аналізу.

Тестування охоплює як **тематичні перевірки**, так і **комплексний підсумковий тест**. Тематичні тести поділені за основними напрямами дисципліни: основи моделювання, об’єктно-орієнтоване моделювання, UML-діаграми, типи зв’язків, життєвий цикл моделі, методи та інструменти. Кожен з них включає по 10 запитань. Підсумковий тест генерується випадково й містить 25 запитань з усіх тем, що дозволяє всебічно перевірити рівень засвоєння матеріалу.

Усі запитання мають чотири варіанти відповідей, серед яких лише один є правильним. Для підвищення ефективності тестування реалізовано розподіл за рівнями складності: легкий (визначення, терміни), середній (застосування знань), складний (аналітичні ситуації). Користувач може переходити між питаннями, змінювати відповіді до завершення тесту, а після проходження — отримує повний звіт з поясненнями до кожної відповіді.

З технічної точки зору, система тестування написана повністю на JavaScript без використання сторонніх бібліотек, що забезпечує легкість, швидкодію та автономність роботи без потреби в серверній частині. Усі дії — вибір відповіді, навігація, підрахунок результатів — виконуються на клієнтській стороні.

Після завершення тесту користувач бачить свій результат у відсотках, кількість правильних і неправильних відповідей, а також пояснення до кожного запитання. Додатково реалізовано механізм збереження статистики тестування

в LocalStorage, що дозволяє переглядати прогрес і порівнювати результати при повторних проходженнях.

Інтерфейс тестування є простим та адаптивним. Він містить прогрес-бар, який демонструє відсоток проходження, кнопки для перемикання між питаннями, а також модальне вікно з результатами після завершення. Оформлення витримано в єдиному стилі з іншими модулями платформи, що забезпечує цілісне сприйняття продукту.

Таким чином, реалізована система тестування є не лише функціональним елементом контролю знань, а й інструментом зворотного зв’язку та самонавчання, що сприяє глибшому засвоєнню матеріалу.

4.5. Інструкція для користувача

У цьому розділі подано покрокову інструкцію з користування веб-платформою «Система контролю знань з моделювання об'єктів». Інтерфейс сайту реалізований таким чином, щоб забезпечити простоту навігації, доступ до навчального контенту та зручне проходження тестів.

1. Головна сторінка (див. рис. 4.4)

Після запуску платформи відкривається головна сторінка з назвою дисципліни, описом модуля, а також кнопками «Почати навчання» та «Пройти тест». Нижче представлено блок «Особливості платформи», що інформує про ключові функції системи: навчальний матеріал, інтерактивне тестування, відстеження прогресу та адаптивний дизайн.

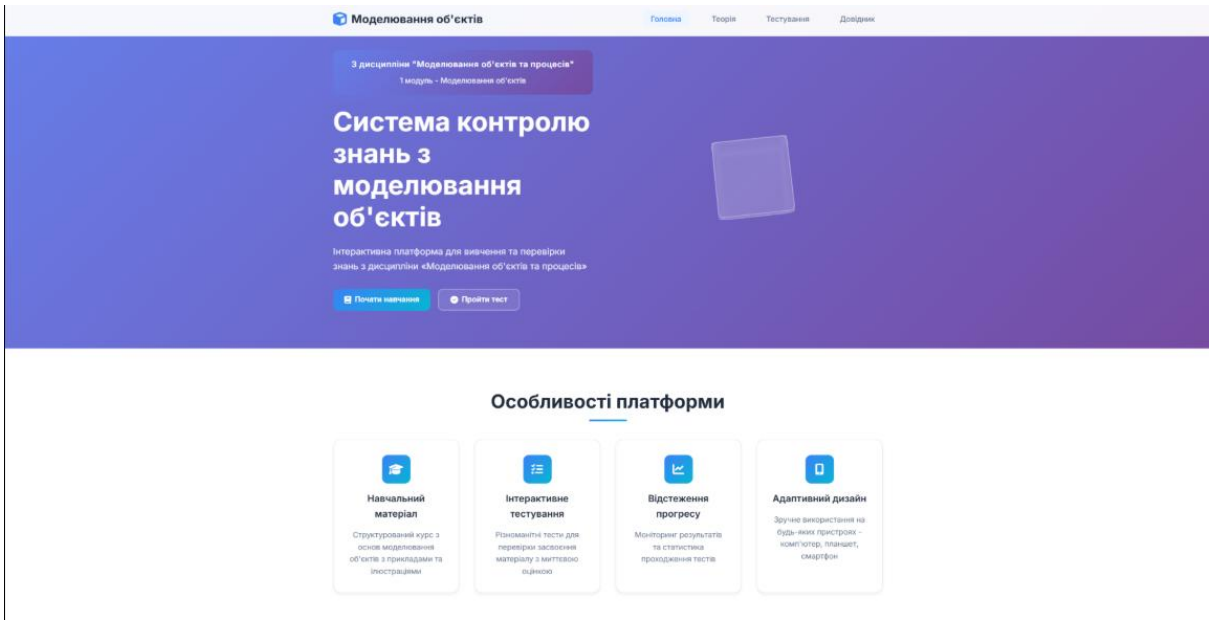

Рисунок 4.4 – Головна сторінка

2. Теоретичний матеріал (див. рис. 4.5)

У розділі «Теорія» користувач може обрати одну з дев’яти тем для опрацювання. Зверху відображається індикатор прогресу вивчення матеріалу. Натиснувши на тему, відкривається докладний текст з виділеними ключовими словами, прикладами та поясненнями. Деякі елементи оформлено у вигляді розгортання (accordion), що дозволяє зручно переглядати підтематику.

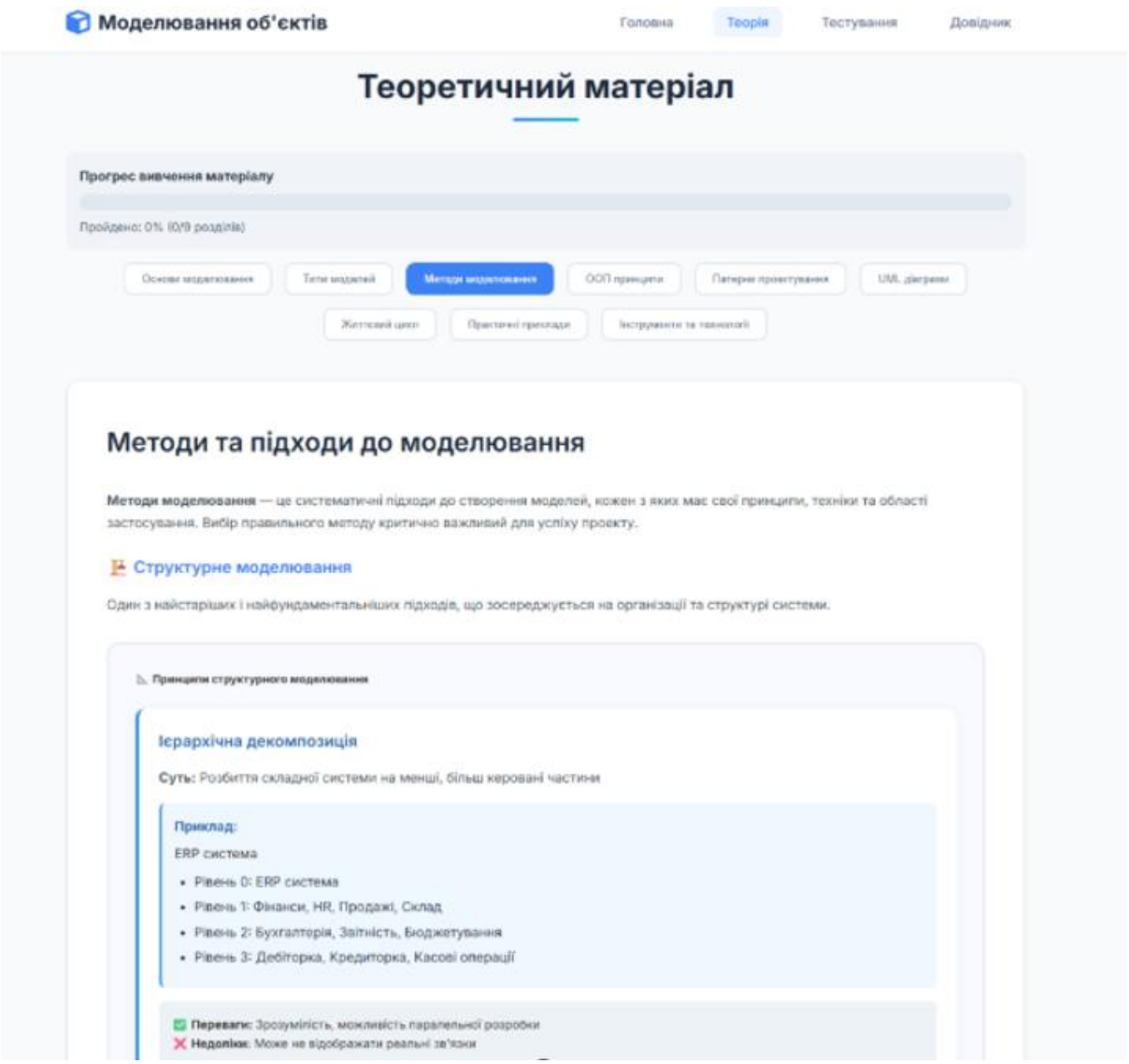

Рисунок 4.5 – Теоретичний матеріал

3. Тематичне тестування (див. рис. 4.6)

У розділі «Тестування» користувач може обрати одну з тематичних категорій. Кожна картка тесту містить назву теми, короткий опис, кількість запитань і орієнтовний час проходження. Такий підхід дозволяє сфокусуватися на конкретній темі та самостійно планувати час для навчання.

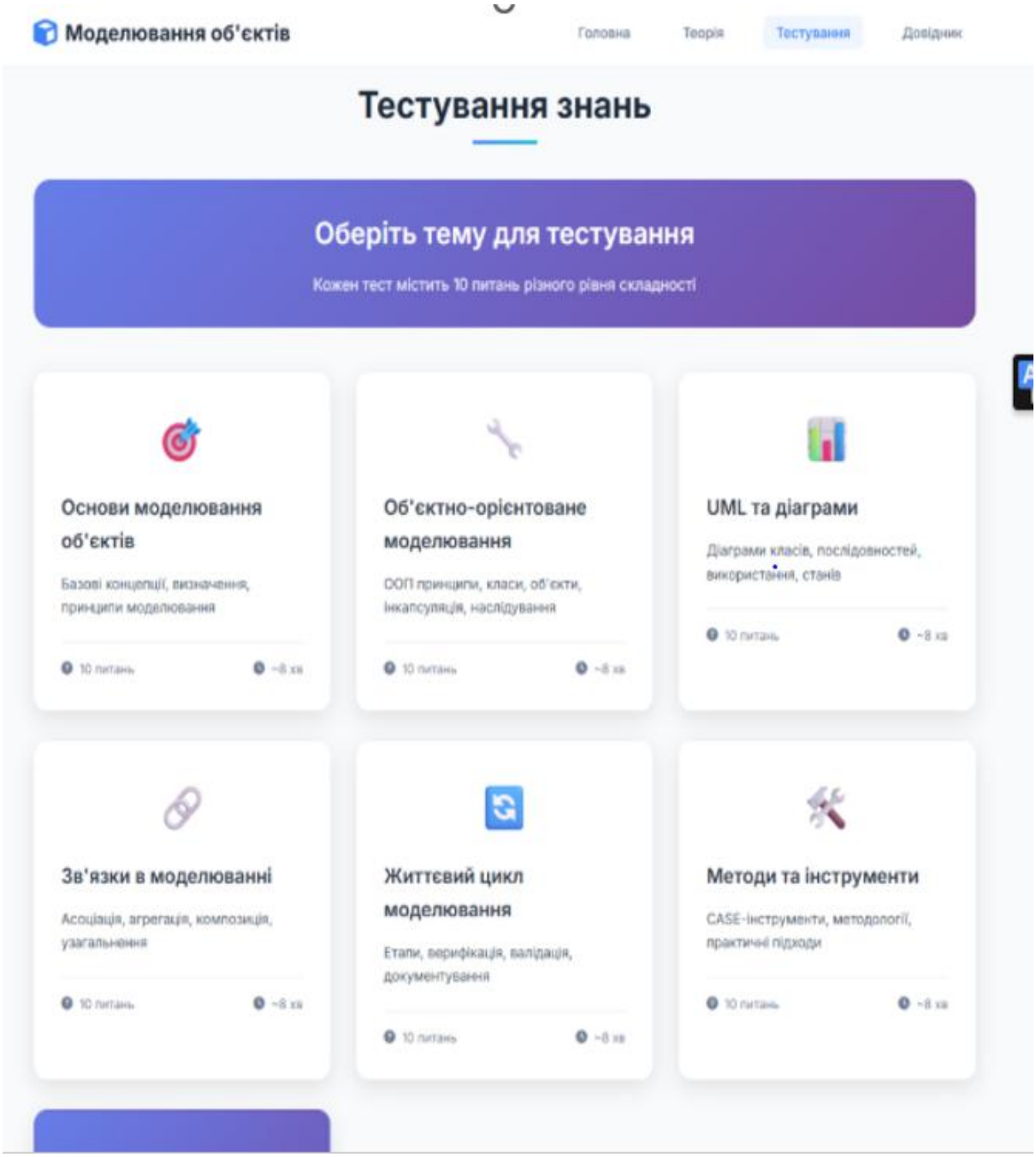

Рисунок 4.6 – Тестування знань

4. Проходження тесту (див. рис. 4.7)

Після вибору теми відкривається інтерфейс проходження тесту. Користувач послідовно відповідає на запитання, маючи змогу перемикатися між ними. Індикатор у верхній частині сторінки показує прогрес, а кольорові мітки вказують на рівень складності запитання.

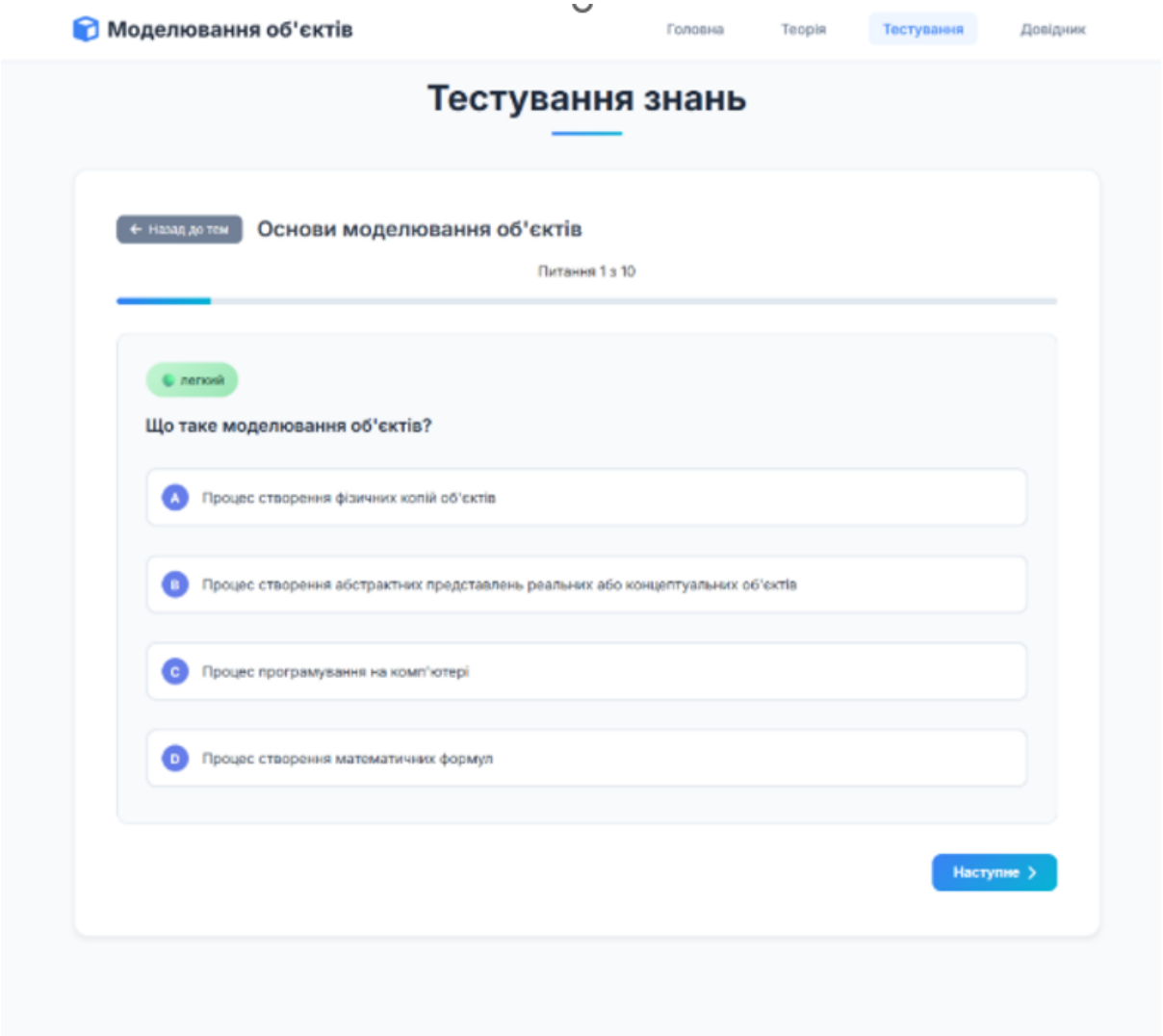

Рисунок 4.7 – Проходження тесту

5. Перегляд результатів (див. рис. 4.8)

Після завершення тесту система формує звіт з результатами: відсоток правильних відповідей, кількість балів, рівень знань. Також подано пояснення до кожної відповіді, що дозволяє користувачеві проаналізувати свої помилки та повторити відповідний розділ теорії.

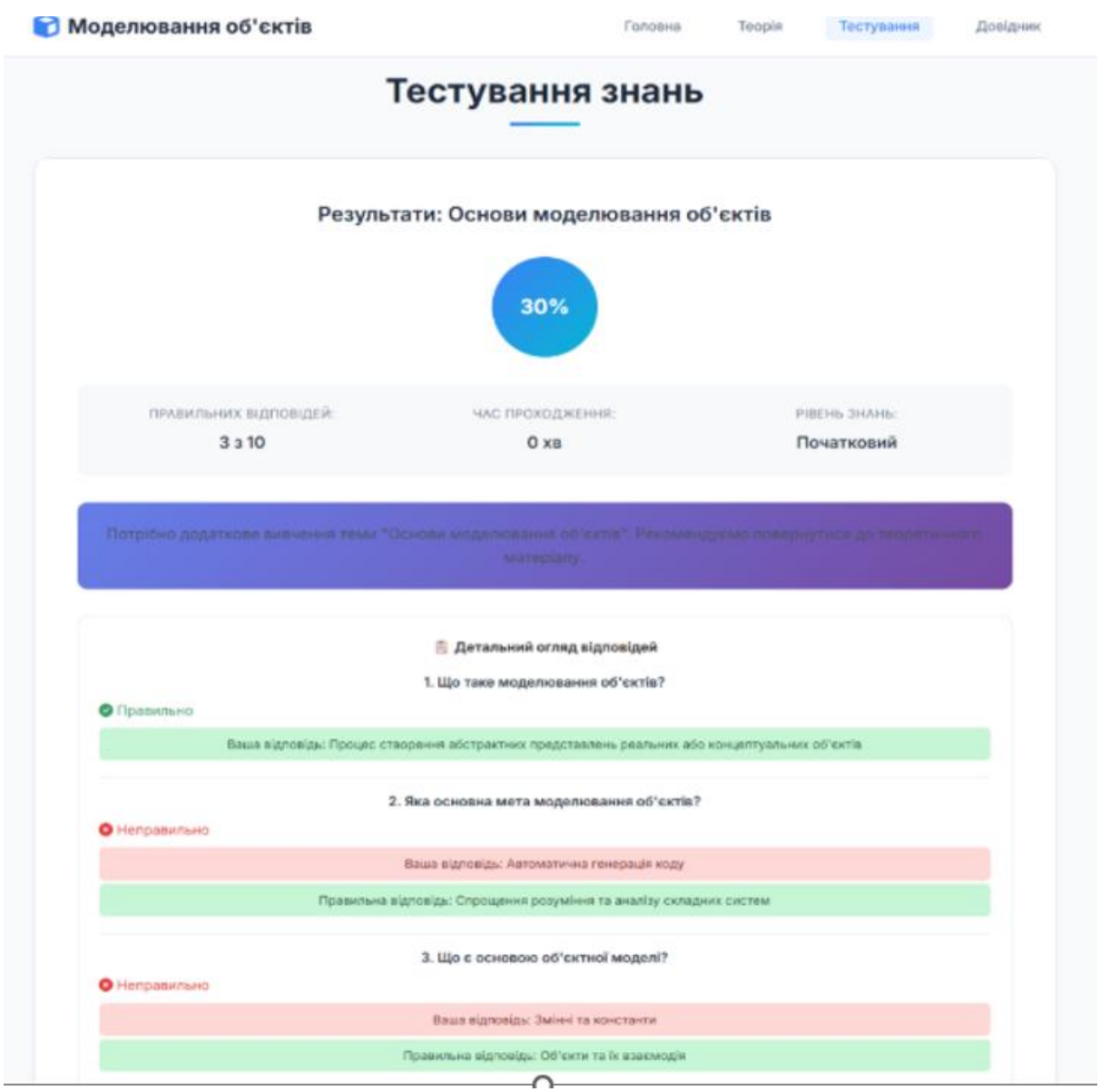

Рисунок 4.8 – Сторінка з результатами

6. Довідник термінів (див. рис. 4.9)

У розділі «Довідник» реалізовано пошук та фільтрацію термінів. Кожен термін супроводжується визначенням, прикладом, а також переліком пов’язаних понять. Завдяки цьому користувач може швидко уточнити значення терміна під час вивчення або проходження тестів.

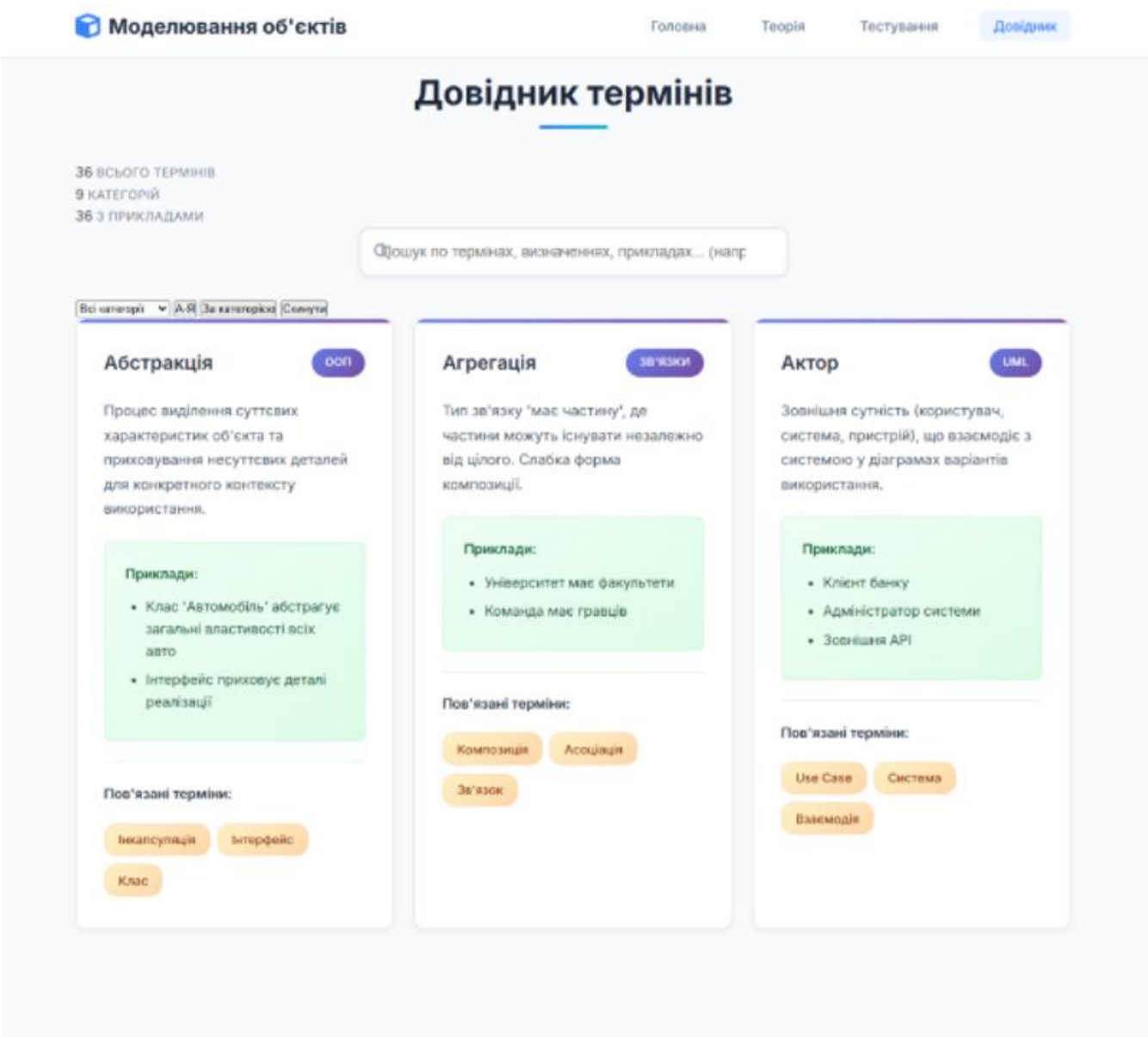

Рисунок 4.9 – Сторінка з довідником

Інтерфейс системи є інтуїтивно зрозумілим, мінімалістичним та адаптованим до будь-яких пристроїв. Користувач може зручно переходити між розділами, працювати з теоретичним матеріалом, проходити тести та переглядати результати без потреби у сторонній допомозі. Це дозволяє ефективно використовувати платформу як для самонавчання, так і для контрольних перевірок у навчальному процесі.

4. Тестування системи.

Тестування є важливим етапом розробки будь-якої програмної системи, оскільки воно дозволяє виявити помилки та недоліки, які можуть вплинути на коректність роботи програми. У цьому розділі розглядаються основні види тестування, які були проведені для системи тестування знань: модульне тестування, інтеграційне тестування та тестування користувацького інтерфейсу.

Модульне тестування спрямоване на перевірку окремих компонентів програми на коректність їх роботи. У контексті розробленої системи тестування знань модульне тестування включало перевірку роботи класів та методів, які відповідають за взаємодію з базою даних, обробку результатів тестування та логіку проходження тесту.

Для проведення модульного тестування було використано фреймворк Google Test, який дозволяє створювати тестові випадки та автоматично перевіряти їх виконання. Наприклад, для перевірки роботи класу DatabaseManager було створено тестовий випадок, який перевіряє коректність підключення до бази даних та виконання SQL-запитів.

Лістинг 3.5

```
#include <gtest/gtest.h>
#include "DatabaseManager.h"

TEST(DatabaseManagerTest, ConnectionTest) {
    DatabaseManager dbManager(":memory:");
    EXPECT_TRUE(dbManager.isConnected());}
TEST(DatabaseManagerTest, QueryExecutionTest) {
    DatabaseManager dbManager(":memory:");
```

QSqlQuery query = dbManager.executeQuery("CREATE TABLE Тести (id INTEGER PRIMARY KEY, назва TEXT)");

EXPECT_FALSE(query.lastError().isValid());}

У першому тесті перевіряється, чи успішно підключено до бази даних. У другому тесті перевіряється коректність виконання SQL-запиту на створення таблиці. Якщо тести проходять успішно, це свідчить про те, що клас DatabaseManager працює коректно.

Також було проведено тестування методу calculateScore, який відповідає за обчислення оцінки користувача на основі правильних та неправильних відповідей. Тестовий випадок перевіряє, чи правильно обчислюється відсоток правильних відповідей.

Лістинг 3.5

```
TEST(ScoreCalculationTest, CalculateScoreTest) {
   QVector<int> userAnswers = {1, 2, 3, 4};
   QVector<int> correctAnswers = {1, 2, 4, 3};
   int score = calculateScore(userAnswers, correctAnswers);
   EXPECT_EQ(score, 50); // Очікуваний результат: 50%}
```

Цей тест перевіряє, чи правильно обчислюється оцінка, якщо користувач дав дві правильні відповіді з чотирьох.

Інтеграційне тестування спрямоване на перевірку взаємодії між окремими компонентами системи. У контексті розробленої системи інтеграційне тестування включало перевірку взаємодії між модулем роботи з базою даних, модулем тестування та модулем перегляду результатів.

Для проведення інтеграційного тестування було створено тестовий сценарій, який імітує повний цикл роботи системи: від вибору тесту до збереження результатів у базі даних. Наприклад, перевірялося, чи коректно зберігаються результати тестування у базі даних після завершення тесту.

Лістинг 3.6

```
TEST(IntegrationTest, SaveResultTest) {
   DatabaseManager dbManager(":memory:");
```

```
dbManager.executeQuery("CREATE TABLE Результати (id INTEGER PRIMARY KEY, ім'я_користувача TEXT, ідентифікатор_тесту INTEGER, дата_та_час TEXT, отримана_оцінка INTEGER)");
QString userName = "Тестовий Користувач";
int testId = 1;
QDateTime dateTime = QDateTime::currentDateTime();
int score = 75;
saveResult(userName, testId, dateTime, score);
QSqlQuery query = dbManager.executeQuery("SELECT * FROM Результати");
EXPECT_TRUE(query.next());
EXPECT_EQ(query.value("ім'я_користувача").toString(), userName);
EXPECT_EQ(query.value("отримана_оцінка").toInt(), score);}
```

Цей тест перевіряє, чи коректно зберігаються дані про результати тестування у базі даних. Якщо тест проходить успішно, це свідчить про те, що модуль роботи з базою даних та модуль обробки результатів працюють коректно разом.

Тестування користувацького інтерфейсу спрямоване на перевірку зручності та коректності взаємодії користувача з системою. У контексті розробленої системи тестування знань тестування інтерфейсу включало перевірку коректності відображення питань, варіантів відповідей, таймера та результатів тестування.

Для проведення тестування інтерфейсу було використано підхід ручного тестування, оскільки автоматизація тестування графічного інтерфейсу у Qt потребує додаткових зусиль. Під час ручного тестування було перевірено наступні аспекти:

- Коректність відображення питань та варіантів відповідей. Перевірялося, чи правильно відображається текст питання та варіанти відповідей на екрані.

- Робота таймера. Перевірялося, чи коректно відображається час, що залишився до завершення тесту, та чи зупиняється тест після закінчення часу.
- Навігація між питаннями. Перевірялося, чи коректно працюють кнопки "Попереднє питання" та "Наступне питання".
- Відображення результатів. Перевірялося, чи коректно відображаються результати тестування, включаючи правильні та неправильні відповіді.

Під час тестування інтерфейсу було виявлено кілька незначних помилок, які були оперативно виправлені. Наприклад, у деяких випадках текст питання не вміщувався на екрані через завелику довжину. Ця проблема була вирішена шляхом додавання можливості прокрутки тексту.

Тестування системи тестування знань показало, що всі основні компоненти програми працюють коректно. Модульне тестування підтвердило правильність роботи окремих класів та методів. Інтеграційне тестування довело, що модулі системи взаємодіють між собою без помилок. Тестування користувацького інтерфейсу забезпечило зручність та інтуїтивність взаємодії користувача з системою. Усі виявлені під час тестування помилки були виправлені, що дозволило забезпечити стабільну та надійну роботу системи.

ВИСНОВКИ

У процесі виконання дипломної роботи було розроблено систему тестування знань, яка дозволяє проводити оцінювання рівня засвоєння студентами матеріалу з модуля "Моделювання об'єктів". Основна увага приділялася проектуванню ключових компонентів системи, зокрема взаємодії з базою даних, реалізації алгоритмів обробки відповідей користувачів. В ході розробки було створено окремі модулі для роботи з базою даних, проходження тестування та аналізу отриманих результатів. Було впроваджено механізм автоматичного розрахунку оцінки на основі відповідей користувача, що дозволяє швидко та об'єктивно оцінювати рівень знань.

Досягнення поставленої мети підтверджується розробкою повноцінного програмного комплексу, який відповідає всім визначеним функціональним вимогам. В результаті тестування було підтверджено коректність роботи основних компонентів системи, включаючи модульну, інтеграційну та користувацьку взаємодію. Всі виявлені під час тестування помилки були усунені, що дозволило забезпечити стабільну та надійну роботу програмного продукту. Система успішно реалізує поставлені завдання, а також має зручний та інтуїтивний інтерфейс для користувачів.

Перспективи подальшого розвитку системи включають розширення її функціональності шляхом впровадження додаткових типів питань, таких як тести з вибором кількох правильних варіантів, завдання на встановлення відповідності та впровадження відкритих запитань. Також доцільним є інтеграція можливості ведення аналітики результатів тестування для викладачів, що дозволить отримувати статистичні дані про успішність студентів у динаміці. Крім того, можливим напрямом розвитку є адаптація системи для роботи в режимі онлайн із збереженням результатів у віддаленій базі даних, що сприятиме її використанню в дистанційному навчанні. Впровадження підтримки мобільних платформ також могло б підвищити доступність системи для користувачів, розширивши її сферу застосування.

Таким чином, проведене дослідження та реалізація програмного продукту дозволили створити ефективну та зручну систему тестування знань. Вона має значний потенціал для подальшого вдосконалення та може бути використана як основа для розробки більш масштабних освітніх платформ.

Список використаної літератури

1. .Методичні рекомендації щодо виконання кваліфікаційної роботи студентів спеціальності 122 Комп’ютерні науки освітня програма «Комп’ютерні науки» ступеня магістра / О. В. Ольховська. – Полтава : ПУЕТ, 2025. – 68 с
2. Грищенко О. О. Основи баз даних. Проектування та управління. – Львів: Видавництво "Магнолія", 2021. – 368 с.
3. Гоувер Дж. SQL та реляційні бази даних. Практичне керівництво. – Харків: Видавництво "Фактор", 2020. – 448 с.
4. SQLite Documentation. Official site: https://www.sqlite.org/docs.html\
5. MDN Web Docs. HTML: HyperText Markup Language. — https://developer.mozilla.org/en-US/docs/Web/HTML
6. W3Schools. Learn JavaScript. — https://www.w3schools.com/js/
7. Google Fonts. Inter — modern sans-serif font. — https://fonts.google.com/specimen/Inter
8. FreeCodeCamp. How to Build Responsive Layouts with CSS Grid. — https://www.freecodecamp.org/news/css-grid-tutorial/
9. GoogleTest Framework Documentation. Official site: https://google.github.io/googletest/
10. Войцех К. Програмне тестування. Теорія та практика. – Київ: Видавництво "Логос", 2021. – 384 с.
11. Стельнік А. І. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТРЕНАЖЕРУ З ТЕМИ «ПРОТОКОЛИ КОМП'ЮТЕРНИХ МЕРЕЖ» ДИСТАНЦІЙНОГО КУРСУ «ІНФОРМАЦІЙНІ МЕРЕЖІ». 2021. URL: http://dspace.puet.edu.ua/handle/123456789/10306
12. Таненбаум Е. Основи комп'ютерних систем. Архітектура, операційні системи та мережі. – Київ: Видавництво "Мір", 2022. – 672 с.
13. IEEE Std 829-2008. Standard for Software and System Test Documentation. – IEEE Computer Society, 2008. – 56 p.
14. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. – Addison-Wesley, 1994. – 395 p.

ДОДАТКИ

Додаток А. Схема бази даних.

```
    }

    ~DatabaseManager() {
        db.close();
    }

    QSqlQuery executeQuery(const QString& query) {
        QSqlQuery q;
        if (!q.exec(query)) {
            qDebug() << "Помилка виконання запиту:" << q.lastError().text();
        }
        return q;
    }

private:
    QSqlDatabase db;
};
```

Реалізація інтерфейсу користувача

```
#include <QWidget>
#include <QLabel>
#include <QPushButton>
#include <QVBoxLayout>
#include <QTimer>

class TestWindow : public QWidget {
    Q_OBJECT

public:
    TestWindow(QWidget *parent = nullptr) : QWidget(parent) {
```

```
        setupUI();
        setupTimer();
    }

private:
    QLabel *questionLabel;
    QPushButton *answerButtons[4];
    QPushButton *prevButton;
    QPushButton *nextButton;
    QLabel *timerLabel;
    QTimer *timer;
    int timeLeft;

    void setupUI() {
        QVBoxLayout *layout = new QVBoxLayout(this);

        questionLabel = new QLabel("Питання буде тут", this);
        layout->addWidget(questionLabel);

        for (int i = 0; i < 4; ++i) {
            answerButtons[i] = new QPushButton("Варіант " + QString::number(i
+ 1), this);
            layout->addWidget(answerButtons[i]);
        }

        prevButton = new QPushButton("Попереднє питання", this);
        nextButton = new QPushButton("Наступне питання", this);
        layout->addWidget(prevButton);
        layout->addWidget(nextButton);
```

```
        timerLabel = new QLabel("Час: 00:00", this);
        layout->addWidget(timerLabel);

        setLayout(layout);
    }

    void setupTimer() {
        timer = new QTimer(this);
        timeLeft = 600; // 10 хвилин
        connect(timer, &QTimer::timeout, this, &TestWindow::updateTimer);
        timer->start(1000); // Оновлення кожну секунду
    }

    void updateTimer() {
        --timeLeft;
        int minutes = timeLeft / 60;
        int seconds = timeLeft % 60;
        timerLabel->setText(QString("Час: %1:%2").arg(minutes, 2, 10,
QLatin1Char('0')).arg(seconds, 2, 10, QLatin1Char('0')));
        if (timeLeft == 0) {
            timer->stop();
            emit testFinished();
        }
    }
};
```

Логіка обробки результатів тестування

```
int calculateScore(const QVector<int>& userAnswers, const QVector<int>&
correctAnswers) {
    int score = 0;
```

```
        for (int i = 0; i < userAnswers.size(); ++i) {
            if (userAnswers[i] == correctAnswers[i]) {
                ++score;
            }
        }
        return (score * 100) / userAnswers.size();
    }
```

Збереження результатів у базі даних

```
    void saveResult(const QString& userName, int testId, const QDateTime&
dateTime, int score) {
        QSqlQuery query;
        query.prepare("INSERT INTO Результати (ім'я_користувача,
ідентифікатор_тесту, дата_та_час, отримана_оцінка) "
                      "VALUES (:userName, :testId, :dateTime, :score)");
        query.bindValue(":userName", userName);
        query.bindValue(":testId", testId);
        query.bindValue(":dateTime", dateTime.toString(Qt::ISODate));
        query.bindValue(":score", score);
        if (!query.exec()) {
            qDebug() << "Помилка збереження результату:" <<
query.lastError().text();
        }
    }
```

Додаток Г. Результати тестування.

Тип тестування	Назва тесту	Опис	Результат	Примітки
Модульне тестування	Тест підключення до бази даних	Перевірка, чи успішно підключено до бази даних	Успішно	
Модульне тестування	Тест виконання SQL-запиту	Перевірка, чи коректно виконується SQL-запит на створення таблиці	Успішно	
Модульне тестування	Тест обчислення оцінки	Перевірка, чи правильно обчислюється оцінка користувача на основі правильних і неправильних відповідей	50% (успішно)	Користувач дав 2 правильні відповіді з 4
Інтеграційне тестування	Тест збереження результатів тесту	Перевірка, чи коректно зберігаються результати тестування в базі даних після завершення тесту	Успішно	
Тестування користувацького інтерфейсу	Перевірка відображення питань та відповідей	Перевірка, чи коректно відображаються питання та варіанти відповідей на екрані	Успішно	
Тестування користувацького інтерфейсу	Перевірка роботи таймера	Перевірка, чи коректно відображається час, що залишився до завершення тесту	Успішно	
Тестування користувацького інтерфейсу	Перевірка навігації між питаннями	Перевірка, чи працюють кнопки "Попереднє питання" та "Наступне питання"	Успішно	